ELSEVIER Journal of Computational and Applied Mathematics 60 (1995) 77–100

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

Parallel homotopy algorithm for symmetric large sparse eigenproblems☆

Liang Jiao Huang[a], Tien-Yien Li[b,*]

[a]*Department of Mathematics, Rockford College, Rockford, IL 61108, United States*
[b]*Department of Mathematics, Michigan State University, East Lansing, MI 48824-1027, United States*

Received 30 September 1993; revised 25 March 1994

Abstract

In this paper, the homotopy continuation method is applied to solve the eigenproblem

$$Ax = \lambda x, \quad \lambda \in \mathbb{R}, \ x \in \mathbb{R}^n \setminus \{0\}$$

for a symmetric large sparse matrix A. A one-parameter family of matrices $A(t) = tA + (1-t)D$ is introduced and the eigenproblem

$$A(t)x(t) = \lambda(t)x(t)$$

is considered for $t \in [0,1]$. We discuss the problem of choosing an optimal starting matrix $A(0) = D$ and consider the regularity and bifurcation problem of $\lambda(t)$ and $x(t)$. A homotopy continuation algorithm is constructed and implemented on both parallel and vector machines for several types of matrices. The numerical experiments show that our method is efficient and highly parallel.

Keywords: Eigenproblems; Homotopy algorithm; Parallelism

1. Introduction

The homotopy method has been successfully applied to solving eigenproblems of both symmetric matrices [15] and unsymmetric matrices [19]. In this paper we consider the computation of eigenvalues and eigenvectors of symmetric large sparse matrices.

Large scale scientific computing is currently an active research field. Traditional methods which work well for small problems are often not suitable for large ones, or not suitable for modern

☆ This research was supported in part by NSF under Grant CCR-9024840.
* Corresponding author.

SSDI 0377-0427(94)00085-9

computer architectures. For example, probably the most efficient and widely used method for solving eigenproblems of small matrices — the QR iteration method [25] — becomes inapplicable for large sparse eigenvalue problems because, among other things, the process of Householder reduction can quickly destroy the sparse pattern. Moreover, the method is highly serial in nature, it is difficult to benefit from advanced architectures. The best known method for large scale eigenvalue problems is the Lanczos method [4]. It can take advantage of the sparseness structure of a given matrix and is good for finding a few extreme eigenvalues. However, it is hard to parallelize and not so efficient for finding interior eigenvalues. Such problems also exist in matrices with special structures that can be efficiently solved by traditional methods [22].

In this paper, the homotopy method is applied to the symmetric large sparse eigenvalue problem. The basic idea is described as follows (see [17, 18] for more details): Given a real symmetric matrix A of order n, consider the one-parameter family of matrices

$$A(t) = D + t(A - D) \quad \text{for } t \in [0,1] \tag{1}$$

with a chosen matrix D. This family has the property that

$$A(0) = D, \qquad A(1) = A.$$

For each $t \in [0,1]$, let the eigenvalues of $A(t)$ be $\lambda_1(t)$, $\lambda_2(t), \ldots, \lambda_n(t)$ such that $\lambda_1(t) \leqslant \lambda_2(t) \leqslant \cdots \leqslant \lambda_n(t)$. It is clear that each $\lambda_i(t)$ is a continuous function of t for $i = 1, 2, \ldots, n$. The graphs of these functions are called *eigencurves*.

Suppose a matrix D is chosen such that its eigensystems are easy to find, then $\lambda_1(0), \lambda_2(0), \ldots, \lambda_n(0)$ are available. By following these continuous eigencurves, the ending points $\lambda_1(1), \lambda_2(1), \ldots, \lambda_n(1)$ of the curves can be reached and the eigenvalues of the given matrix A are found.

The main advantages of this method are:

- It is parallel in nature: the tracing of one eigencurve is independent of those of the others.
- The main calculation is concentrated on solving large sparse linear equations, and unlike solving for eigenvalues directly, several software packages already exist for solving such linear equations efficiently [7, 8, 10].
- It can be used to find only a few of the specified eigenvalues. In contrast, the Lanczos method tends to give extreme eigenvalues on both ends before the emerging of interior eigenvalues, and it does not identify the index and multiplicity of a computed eigenvalue.

Our paper proceeds as follows: Section 2 discusses the problem of choosing a starting matrix D. Some optimal solutions to this question are found under certain conditions. Section 3 addresses the regularity and bifurcation problems. Under the assumption that A and D are symmetric, the eigensystems are analytic and the bifurcations are easy to handle — bifurcation directions can be readily computed using a simple formula. Section 4 describes our homotopy algorithm for following the eigencurves. Section 5 presents the test matrices, softwares used, and the numerical results from a vector machine and a parallel machine. The results show that the homotopy method is efficient and highly scalable for large sparse eigenproblems.[1]

[1] Our code is available upon request (e-mail: lhuang@aol.com). Our code calls subroutines from the commercial package "Harwell Subroutine Library", such as subroutines for solving large sparse linear system of equations and Lanczos algorithm.

2. The choice of starting matrix

To construct our homotopy, i.e., the parameterized matrix family $A(t)$, a starting matrix D must be chosen first. The efficiency of our algorithm depends heavily on the choice of D. First of all, D must be a simple matrix, so that its eigensystem can be obtained much easier than that of A. Secondly, D needs to be as close to A as possible, so that the eigencurves generated are easy to follow (in the extreme case, if $D = A$, all curves become straight lines!).

Unless more is known about A, the block diagonal matrix will be our candidate,

$$D = \begin{bmatrix} D_{11} & & & \\ & D_{22} & & \\ & & \ddots & \\ & & & D_{kk} \end{bmatrix}, \tag{2}$$

where D_{ii} are smaller square matrices for $i = 1, 2, \ldots, k$. For such a matrix D, its eigensystems can be trivially assembled from those of D_{ii}'s, and each D_{ii}'s eigensystems can be easily found since it is a small matrix. Furthermore, by using multi-processors, eigensystems of D can be computed in parallel. The natural question is then how to find a D of this form which is closest to A. In other words, let $\mathscr{D}$ be the set of matrices of the form (2), our goal is to find a $D_0 \in \mathscr{D}$ such that

$$\| A - D_0 \| = \min_{\mathscr{D}} \| A - D \|. \tag{3}$$

The unitarily invariant norms are used to measure the closeness of two matrices. Two important classes of unitarily invariant norms are: *Schatten p norms*

$$\| A \|_p = \left(\sum_{j=1}^{n} s_j(A)^p \right)^{1/p}, \quad p \geqslant 1$$

and *Ky Fan k norms*

$$\| A \|_k = \sum_{j=1}^{k} s_j(A), \quad k = 1, 2, \ldots, n.$$

where $\{s_j(A), j = 1, 2, \ldots, n\}$ are the singular values of A in descending order. Schatten 2 norm is the Frobenius norm $\| \cdot \|_F$ and Ky Fan 1 norm is the spectral norm $\| \cdot \|_2$.

For the Frobenius norm, the solution to (3) is obvious. We can partition A into blocks according to the structure in (2) and choose D_0 to be the block diagonal matrix whose diagonal blocks are identical to the corresponding ones of A. That is, if

$$A = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1k} \\ A_{21} & A_{22} & \cdots & A_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ A_{k1} & A_{k2} & \cdots & A_{kk} \end{bmatrix}, \tag{4}$$

then

$$D_0 = \begin{bmatrix} A_{11} & & & \\ & A_{22} & & \\ & & \ddots & \\ & & & A_{kk} \end{bmatrix}. \tag{5}$$

For general unitarily invariant norms, however, the solution is not so obvious. One would hope that D_0 above is still the solution. It turns out that this is true for an important class of matrices but not for all matrices.

In this section, some positive results are established first, then a counter example is constructed to show that the result cannot be extended to cover all matrices.

Theorem 2.1. *Let A and D_0 be matrices in $\mathbb{C}^{n\times n}$ as in* (4) *and* (5). *If there exists a unitary matrix $E \in \mathscr{U}$ which commutes with every $D \in \mathscr{D}$ such that*

$$A - D_0 = -E(A - D_0)E^{\mathrm{H}}, \tag{6}$$

then D_0 is the best block diagonal approximant of A for every unitarily invariant norm, i.e., $\|A - D_0\| = \min_{D\in\mathscr{D}} \|A - D\|$. Here E^{H} represents the Hermitian transpose of E.

An important class of matrices satisfying (6) is the class of block tridiagonal matrices.

Corollary 2.2. *If A is a block tridiagonal matrix, then the best block diagonal approximant of A is D_0 for every unitarily invariant norm.*

Proof. Condition (6) is satisfied for $E = \mathrm{diag}(I_1, -I_2, \ldots, (-1)^{k-1}I_k)$, with each I_j an identity matrix of appropriate order. □

Block tridiagonal (in particular, tridiagonal) matrices arise in many applications. In fact, in matrix eigenvalue computations, a given matrix is usually transformed into a compact form — tridiagonal or block tridiagonal form by Householder or Lanczos transformation [4], then QR iteration, QR/INVIT method, bisection, or homotopy method can be applied to obtain the final solutions.

For general matrices, we have the following corollary.

Corollary 2.3. *Let A be any square matrix, if we partition A into*

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix},$$

then

$$D_0 = \begin{bmatrix} A_{11} & 0 \\ 0 & A_{22} \end{bmatrix}$$

is the best approximant of A for every unitarily invariant norm.

Proof. This is a special case of Corollary 2.2 with $k = 2$. □

This result has its own significance. It is well known that the time required for finding eigensystems of a matrix of order n is proportional to n^3. If A is divided approximately equally into two blocks as in Corollary 2.2, then D_0 is closest to A and the eigensystems of D can be solved by parallel computer using about 1/8 of the execution time for A. Furthermore, if the block sizes of D_0 are still too large to work with, they can be divided into smaller blocks and the "divide and conquer" strategy can be employed.

For the proof of Theorem 2.1, the following lemmas are needed. They were first proved in [9].

Lemma 2.4. *Let $A, B \in \mathcal{M}$, then $\|A\| \leqslant \|B\|$ holds for all unitarily invariant norms $\|\cdot\|$ if and only if*

$$\|A\|_k \leqslant \|B\|_k$$

holds for all Ky Fan norms $\|\cdot\|_k$, $k = 1, 2, \ldots, n$.

Lemma 2.5. *Let $A \in \mathbb{C}^{n\times n}$, $X_k = \{x_1, x_2, \ldots, x_k\}$ and $Y_k = \{y_1, y_2, \ldots, y_k\}$ be sets of orthonormal vectors in $\mathbb{C}^n$, then*

$$\|A\|_k = \sum_{j=1}^{k} s_j(A) = \max_{X_k, Y_k} \sum_{j=1}^{k} \mathrm{Re}\langle x_j, Ay_j\rangle.$$

Proof of Theorem 2.1. Let $A_0 = A - D_0 = U\Sigma V^{\mathrm{H}}$ be a singular value decomposition of A_0, with

$$U = (u_1, u_2, \ldots, u_n), \quad \Sigma = \mathrm{diag}(s_1(A_0), \ldots, s_n(A_0)), \quad V = (v_1, v_2, \ldots, v_n).$$

Because $A_0 = -EA_0E^{\mathrm{H}}$,

$$A_0 = -E(U\Sigma V^{\mathrm{H}})E^{\mathrm{H}} = (-EU)\Sigma(EV)^{\mathrm{H}},$$

where $-EU$ and EV are unitary matrices. This gives another singular value decomposition of A_0. Thus,

$$\langle u_j, A_0 v_j\rangle = \langle -Eu_j, A_0 Ev_j\rangle = s_j(A_0).$$

By Lemma 2.5, for any $D \in \mathcal{D}$ and $k \leqslant n$,

$$\begin{aligned}\|A - D\|_k &\geqslant \sum_{j=1}^{k} \mathrm{Re}\langle u_j, (A - D)v_j\rangle \\ &= \sum_{j=1}^{k} \langle u_j, A_0 v_j\rangle + \sum_{j=1}^{k} \mathrm{Re}\langle u_j, (D_0 - D)v_j\rangle \\ &= \|A - D_0\|_k + \sum_{j=1}^{k} \mathrm{Re}\langle u_j, (D_0 - D)v_j\rangle. \end{aligned} \tag{7}$$

On the other hand, since $-EU$ and EV are unitary matrices,

$$\begin{aligned}\|A-D\|_k &\geqslant \sum_{j=1}^{k} \operatorname{Re}\langle -Eu_j,(A-D)Ev_j\rangle \\ &= \sum_{j=1}^{k} \langle -Eu_j, A_0Ev_j\rangle + \sum_{j=1}^{k} \operatorname{Re}\langle -Eu_j,(D_0-D)Ev_j\rangle \\ &= \sum_{j=1}^{k} s_j(A_0) - \sum_{j=1}^{k} \operatorname{Re}\langle Eu_j, E(D_0-D)v_j\rangle \\ &= \|A-D_0\|_k - \sum_{j=1}^{k} \operatorname{Re}\langle u_j,(D_0-D)v_j\rangle. \end{aligned} \tag{8}$$

Combining inequalities (7) and (8),

$$\begin{aligned}\|A-D\|_k &\geqslant \|A-D_0\|_k + \left|\sum_{j=1}^{k} \operatorname{Re}\langle u_j,(D_0-D)v_j\rangle\right| \\ &\geqslant \|A-D_0\|_k.\end{aligned}$$

This completes the proof because of Lemma 2.4. □

For general matrices, the conclusion of Theorem 2.1 is not true, this can be seen in the following example.

Example 2.6. Let

$$A=\begin{bmatrix}0 & 2 & 2\\ 2 & 0 & 2\\ 2 & 2 & 0\end{bmatrix}.$$

If we choose block size to be 1, then $D_0=0$. The eigenvalues of A are $\lambda_1=4$, $\lambda_2=-2$ and $\lambda_3=-2$, therefore for spectral norm $\|\cdot\|_2$,

$$\|A-D_0\|_2 = \max_{1\leqslant i\leqslant n} |\lambda_i| = \lambda_1 = 4.$$

However,

$$\|A-I\|_2 = \max_{1\leqslant i\leqslant n} |\lambda_i - 1| = 3,$$

i.e.,

$$\|A-I\|_2 < \|A-D_0\|_2.$$

Therefore D_0 is not the best approximant. In fact, it can be shown that I is the best approximant in this case: first of all, $A - I$ has orthonormal eigenvectors

$$x_1 = \frac{1}{\sqrt{3}}\begin{bmatrix}1\\1\\1\end{bmatrix}, \qquad x_2 = \frac{1}{\sqrt{2}}\begin{bmatrix}0\\1\\-1\end{bmatrix}, \qquad x_3 = \frac{1}{\sqrt{6}}\begin{bmatrix}-2\\1\\1\end{bmatrix}$$

with corresponding eigenvalues $\lambda_1 = 3$, $\lambda_2 = -3$ and $\lambda_3 = -3$. If there exists a matrix D such that $\|A - D\|_2 < \|A - I\|_2$, then by the Courant–Fischer min–max theorem,

$$\begin{aligned}|\langle x_j,(A-D)x_j\rangle| &= |\langle x_j,(A-I)x_j\rangle + \langle x_j,(I-D)x_j\rangle| \\ &= |\lambda_j + \langle x_j,(I-D)x_j\rangle| \leqslant \|A-D\|_2 \\ &< \|A-I\|_2 = |\lambda_j|, \quad j = 1,2,3.\end{aligned}$$

Let $I - D = \operatorname{diag}(d_1, d_2, d_3)$, then the above inequalities yield

$$\langle x_1,(I-D)x_1\rangle = \tfrac{1}{3}(d_1 + d_2 + d_3) < 0, \tag{9}$$

$$\langle x_2,(I-D)x_2\rangle = \tfrac{1}{2}(d_2 + d_3) > 0, \tag{10}$$

$$\langle x_3,(I-D)x_3\rangle = \tfrac{1}{6}(4d_1 + d_2 + d_3) > 0. \tag{11}$$

It follows from (9) and (10), $d_1 < 0$. However, (9) and (11) imply $3d_1 > 0$. Thus such a matrix D does not exist, and the assertion is achieved.

Remark 2.7. Our results provide a general guideline for choosing a starting matrix. In a practical problem, usually more special properties about the given matrix A are known, this allows other choices for D as long as it is close to A and its eigensystems are easy to compute. For example, it is often the case that in order to determine the final parameters for a problem, it is necessary to do a series of experiments. In each experiment, the parameters are adjusted by a small amount. If the question is related to an eigenvalue problem, then there is a family of matrices with the same (or similar) structures, each matrix is a small perturbation of the previous one (except the first). In such a case, the natural choice for the starting matrix is one of the matrices in the family whose eigensystems have been found, and it is expected that the eigensystems of the new matrix is a "small" perturbation of the starting matrix. Some numerical results of this type in the last section show that our method is highly efficient for such a situation.

3. Regularity and bifurcation

Before turning to the description of our algorithm, let us consider the regularity and bifurcation of the eigencurves.

It is well known that eigenvalues are continuous functions of the entries of the matrix. However, they are generally not differentiable. For example, consider

$$A(t) = \begin{bmatrix} 1 & t \\ 1 & 1 \end{bmatrix},$$

its eigenvalues are $\lambda_1(t) = 1 - \sqrt{t}$ and $\lambda_2(t) = 1 + \sqrt{t}$. They are not differentiable at $t = 0$. This happens because, at $t = 0$, $A(0)$ has multiple eigenvalues $\lambda_1(0) = \lambda_2(0) = 1$, that is, eigencurves have a bifurcation point.

The behavior of eigenvectors at a bifurcation is even worse. They may not even be continuous. The following matrix [23]

$$A(t) = \begin{bmatrix} 1 + t\cos(2/t) & -t\sin(2/t) \\ -t\sin(2/t) & 1 - t\cos(2/t) \end{bmatrix}$$

has eigenvalues $\lambda_1(t) = 1 - t$ and $\lambda_2(t) = 1 + t$. They are analytic functions of t. However, the corresponding eigenvectors

$$x_1(t) = \begin{bmatrix} \cos(1/t) \\ -\sin(1/t) \end{bmatrix}, \qquad x_2(t) = \begin{bmatrix} \sin(1/t) \\ \cos(1/t) \end{bmatrix},$$

have no limits as $t \to 0$. Again the problem arises because $A(t)$ has a double eigenvalue at $t = 0$.

3.1. *Computation of bifurcations*

Handling the bifurcations efficiently is a very important problem for homotopy algorithms. The behavior of eigencurves around such points can be quite complicated. Fortunately, for Hermitian matrices, bifurcation is not as difficult. In fact, it is known that $\lambda_i(t)$ and $x_i(t)$ can be chosen in such a way that they are all analytic functions of t, making the eigencurves through a bifurcation well conditioned and relatively easy to follow.

Theorem 3.1. *Suppose A and D are both real symmetric matrices. Let*

$$A(t) = tA + (1 - t)D = D + t(A - D),$$

then the eigensystems

$$(\lambda_1(t), x_1(t)), (\lambda_2(t), x_2(t)), \ldots, (\lambda_n(t), x_n(t))$$

of $A(t)$ can be chosen in such a way that all functions involved are analytic functions for real t. Furthermore, there are only finitely many $t \in [0, 1]$ such that $A(t)$ has multiple eigenvalues.

Proof. See [13]. □

Suppose $(\lambda_1(t), x_1(t)), (\lambda_2(t), x_2(t)), \ldots, (\lambda_n(t), x_n(t))$ are the eigensystems of the real symmetric matrix $A(t)$ as in Theorem 3.1, then

$$A(t)x_j(t) = \lambda_j(t)x_j(t). \tag{12}$$

Differentiate both sides with respect to t,

$$A'(t)x_j(t) + A(t)x_j'(t) = \lambda_j'(t)x_j(t) + \lambda_j(t)x_j'(t). \tag{13}$$

Multiplying the above equation on the left by $x_j^{\mathrm{T}}(t)$ yields,

$$\lambda_j'(t) = x_j^{\mathrm{T}}(t)A'(t)x_j(t).$$

But $A'(t) = A - D$, so

$$\lambda_j'(t) = x_j^{\mathrm{T}}(t)(A - D)x_j(t). \tag{14}$$

With this last formula, the prediction–correction scheme can be applied to numerically compute the eigenvalues. However, at a bifurcation point, eigenvectors are not uniquely defined. For an eigenvalue of multiplicity k, any k orthonormal vectors from the k-dimensional invariant subspace form an eigenbasis for the subspace. By Theorem 3.1, there is at least one way for choosing an appropriate set of eigenvectors for each t such that $x_j(t)$ becomes analytic. This choice is not known beforehand. And (14) can only be applied for this set of $x_j(t)$'s. An alternative way of computing these bifurcation directions is given as follows.

Theorem 3.2. *Suppose* $\lambda_i(t) = \lambda_{i+1}(t) = \cdots = \lambda_{i+k-1}(t)$ *are* k *multiple eigenvalues of* $A(t)$. *Let* $y_1(t), y_2(t), \ldots, y_k(t)$ *be any* k *orthonormal eigenvectors corresponding to these eigenvalues. Then* $\lambda_i'(t), \lambda_{i+1}'(t), \ldots, \lambda_{i+k-1}'(t)$ *equal the* k *eigenvalues of* $Y^{\mathrm{T}}(A - D)Y$, *where* Y *is the matrix consisting of* $y_i(t), y_{i+1}(t), \ldots, y_{i+k-1}(t)$ *as its columns.*

Proof. The proof of a more general result can be found in [13]. By using Eq. (13), a very simple proof can be obtained here.

Let $x_i(t), x_{i+1}(t), \ldots, x_{i+k-1}(t)$ be the analytic eigenvectors corresponding to $\lambda_i(t) = \lambda_{i+1}(t) = \cdots = \lambda_{i+k-1}(t)$ as in Theorem 3.1. Multiply (13) by $x_l^{\mathrm{T}}(t)$ from the left,

$$x_l^{\mathrm{T}}(t)(A - D)x_j(t) + \lambda_l(t)x_l^{\mathrm{T}}(t)x_j'(t) = \lambda_j'(t)x_l^{\mathrm{T}}(t)x_j(t) + \lambda_j(t)x_l^{\mathrm{T}}(t)x_j'(t).$$

Since $\lambda_l(t) = \lambda_j(t)$ and $x_l^{\mathrm{T}}(t)x_j(t) = \delta_{lj}$ for $i \leqslant l, j \leqslant i + k - 1$, the above equation yields

$$x_l^{\mathrm{T}}(t)(A - D)x_j(t) = \delta_{lj}\lambda_j'(t) \quad \text{for } i \leqslant l,\ j \leqslant i + k - 1. \tag{15}$$

Let X be the matrix consisting of $x_i(t), x_{i+1}(t), \ldots, x_{i+k-1}(t)$ as its columns. By (15),

$$\Lambda'(t) \equiv X^{\mathrm{T}}(A - D)X = \begin{bmatrix} \lambda_i'(t) & & & \\ & \lambda_{i+1}'(t) & & \\ & & \ddots & \\ & & & \lambda_{i+k-1}'(t) \end{bmatrix}. \tag{16}$$

Because both $\{y_i(t), y_{i+1}(t), \ldots, y_{i+k-1}(t)\}$ and $\{x_i(t), x_{i+1}(t), \ldots, x_{i+k-1}(t)\}$ form orthonormal bases for the invariant subspace, there exists an orthogonal matrix Q such that $Y = XQ$. Hence

$$Y^{\mathrm{T}}(A - D)Y = Q^{\mathrm{T}}[X^{\mathrm{T}}(A - D)X]Q.$$

It follows that $Y^T(A - D)Y$ and $X^T(A - D)X$ are similar matrices and have the same eigenvalues. By (16), they are $\lambda_i'(t), \lambda_{i+1}'(t), \ldots, \lambda_{i+k-1}'(t)$. □

3.2. Continuity of eigenvectors

The Rayleigh quotient iteration (RQI) is one of the most important methods in our algorithm (see Section 4). The convergence of RQI depends not only on a good eigenvalue prediction, but also on a good eigenvector prediction. The following example shows that using the eigenvectors at step t as the starting vector at step $t + \Delta t$ can be an inefficient choice.

Let

$$A = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \qquad D = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

and $A(t) = (1 - t)D + tA$, then

$$A(t) = \begin{bmatrix} 1 & t \\ t & 1 \end{bmatrix}.$$

The computed eigenvectors for $A(0) = D$ are

$$x_1(0) = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad x_2(0) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

However, for any $t > 0$, $A(t)$ has eigenvectors

$$x_1(t) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad x_2(t) = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

These eigenvectors are not even continuous at $t = 0$. In fact there is a 45° turn in both vectors. Using $x_1(0)$ and $x_2(0)$ as starting vectors for RQI in the computation of $x_1(t)$ and $x_2(t)$ is clearly undesirable. Again, the problem occurs because $A(t)$ has multiple eigenvalues at $t = 0$, and any two orthonormal vectors can serve as base vectors for the corresponding invariant subspace. An important question is then, how can one find a set of eigenvectors at step t that can be turned continuously into a set of eigenvectors at step $t + \Delta t$? Using the notation in Theorem 3.2, we have the following theorem.

Theorem 3.3. *Suppose $Y^T(A - D)Y$ has eigendecomposition $Q\Lambda'Q^T$. If*

$$\lambda_i'(t), \lambda_{i+1}'(t), \ldots, \lambda_{i+k-1}'(t)$$

are distinct, then the columns of YQ are, up to a sign, the analytic eigenvectors described in Theorem 3.1.

Proof. Let X be the matrix with the analytic eigenvectors

$$x_i(t), x_{i+1}(t), \ldots, x_{i+k-1}(t)$$

as its columns. Since both X and Y consist of orthonormal eigenvectors from the same invariant subspace, there is an orthogonal matrix $\tilde{Q}$ such that $X = Y\tilde{Q}$. Substituting into (16), yields

$$\tilde{Q}^{\mathrm{T}}(Y^{\mathrm{T}}(A - D)Y)\tilde{Q} = \Lambda',$$

or

$$Y^{\mathrm{T}}(A - D)Y = \tilde{Q}\Lambda'\tilde{Q}^{\mathrm{T}}.$$

On the other hand,

$$Y^{\mathrm{T}}(A - D)Y = Q\Lambda'Q^{\mathrm{T}},$$

so,

$$Q\Lambda'Q^{\mathrm{T}} = \tilde{Q}\Lambda'\tilde{Q}^{\mathrm{T}}.$$

Because the diagonal matrix Λ' has distinct values $\lambda_i'(t), \lambda_{i+1}'(t), \ldots, \lambda_{i+k-1}'(t)$ along its diagonal, the columns of Q and $\tilde{Q}$ can differ at most by a sign. Hence the columns of YQ and $X = Y\tilde{Q}$ can differ at most by a sign. □

In conclusion, when bifurcation occurs at t, $Y^{\mathrm{T}}(A - D)Y$ is formed by using the computed eigenvector matrix Y, then the eigendecomposition $Q\Lambda'Q^{\mathrm{T}}$ is computed for this $k \times k$ matrix $Y^{\mathrm{T}}(A - D)Y$. From this decomposition, the bifurcation directions can be obtained. If these directions are mutually distinct, then YQ generates a set of improved eigenvectors, which can be used to speed up the iterations in the next step.

For the example considered at the beginning of this section, a simple computation gives the improved eigenvectors for all t:

$$x_1(0) = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\1\end{bmatrix}, \qquad x_2(0) = \frac{1}{\sqrt{2}}\begin{bmatrix}1\\-1\end{bmatrix}.$$

They turn out to be the exact eigenvectors at $t = 1$.

4. The algorithm

The tracing of eigencurves consists of the following main steps.

4.1. Choice of the starting matrix

To initiate the algorithm, a starting matrix D must be chosen first. According to the discussion in Section 2, block diagonal matrix with blocks directly from A will be our choice unless more information is available. The appropriate sizes of the blocks were discussed there. It is clear that if diagonal blocks of D are larger, it will be closer to A, and the eigencurves will have better conditions. However, because of the obvious requirement that the eigensystems of D should be much easier to find than those of A, the size must be kept under a certain limit. In our algorithm,

QR iteration is used to solve for eigensystems of D. This requires that block size be no more than a few hundred. Another important factor in choosing D is the computer architecture. If more processors are available, more (hence smaller) blocks can be used. In the test examples we run on the BBN Butterfly Machine, block sizes are around one hundred. Comparing with other choices, it appears that this selection is close to optimal for most of the cases.

4.2. Location of the starting points

Starting points $(\lambda_i(0), x_i(0))$ for $i = 1, 2, \ldots, n$ are eigenpairs of D. Since D is a block diagonal matrix, its eigenvalues consist of eigenvalues from all diagonal blocks, its eigenvectors can be obtained from eigenvectors of diagonal blocks by extending them to full dimension vectors with appropriate zero entries. In a parallel architecture, each block is delivered to a processor and QR iteration is applied. The results are collected to form the eigensystems of D. Finally the eigenvalues of D are arranged in descending order for parallel processing in later steps.

Apparently this step has a high parallelization level and requires a relatively small amount of execution time.

4.3. Prediction

Assuming that for the jth eigencurve the values λ_j and x_j at t are known, we now describe the algorithm to approximate λ_j and x_j at $t + h$. The choice of the step size h will be discussed later. In this step in addition to predicting the values, we also determine if $\lambda_j(t + h)$ is an isolated eigenvalue, because the correction in the next step for an isolated eigenvalue is different from that of a cluster.

Case 1. The jth eigenvalue $\lambda_i(t)$ of $A(t)$ is simple and well isolated from others, then

$$\lambda_j'(t) = x_j^{\mathrm{T}}(t)(A - D)x_j(t).$$

Let the jth predicted eigenvalue of $A(t + h)$ be

$$\begin{aligned}\lambda_j(t + h) &\approx \lambda_j(t) + \lambda_j'(t)h \\ &= \lambda_j(t) + x_j^{\mathrm{T}}(t)(A - D)x_j(t)h.\end{aligned}$$

All quantities on the right-hand side of the above equation are available from step t.

For the eigenvector, simply use $x_j(t)$ as the prediction for $x_j(t + h)$. There are several reasons for doing this:

- Although in theory one can use

$$x_j(t + h) \approx x_j(t) + x_j'(t)h, \tag{17}$$

  but the high cost in computing $x_j'(t)$ makes this choice impractical.
- When $\lambda_j(t)$ is well isolated, $x_j(t)$ is often a good prediction for $x_j(t + h)$; if $\lambda_j(t)$ is not so well isolated, the eigenvector is sensitive to perturbation, the above formula may not yield significant improvement.
- In our correction step, the eigenvector prediction is not as important as the eigenvalue prediction.

Case 2. $\lambda_j(t)$ is in a cluster of eigenvalues of $A(t)$ (this can be monitored during traversing), then we proceed as follows:

(1) find k — the number of eigenvalues in the cluster;
(2) form $X = [x_i, x_{i+1}, \dots, x_{i+k-1}]$ where the corresponding $\lambda_i, \dots, \lambda_{i+k-1}$ form a cluster;
(3) form $X^T(A - D)X$;
(4) calculate eigendecomposition $X^T(A - D)X = Q\Omega Q^T$;
(5) set $X = XQ$.

Let $\Omega = \operatorname{diag}(\omega_i, \omega_{i+1}, \dots, \omega_{i+k-1})$. According to Theorem 3.2, $\omega_i, \omega_{i+1}, \dots, \omega_{i+k-1}$ are derivatives $\lambda_j'(t)$ for $i \leqslant j \leqslant i + k - 1$. Hence we form the predictions:

$$\lambda_j(t + h) \approx \lambda_j(t) + \omega_j h, \tag{18}$$

$$x_j(t + h) \approx x_j(t). \tag{19}$$

4.4. Correction

From the last step, in addition to the predictions for $\lambda_j(t + h)$ and $x_j(t + h)$, the information about the isolation of $\lambda_j(t + h)$ becomes available. If it should be treated as an eigenvalue in a cluster, the number of eigenvalues in the cluster is also known, then the Subspace Iteration with Rayleigh–Ritz Procedure (SIRR) will be used to execute the correction. This algorithm is discussed in detail in [24].

If $\lambda_j(t + h)$ can be treated as an isolated eigenvalue, the correction is done by using the Inverse Iteration (IVIT) followed by Rayleigh Quotient Iteration (RQI). Because of the following two observations, RQI is not applied at the first step:

(a) Although $\lambda_j(t)$ is isolated, $\lambda_j(t + h)$ may not be so, and our prediction step cannot detect such a situation. When this happens, if RQI is used, those undesirable behaviors of the algorithm, such as converging to an eigenvalue far from the original prediction, might occur [21].

(b) Although RQI has a cubic convergence rate, usually a few iterations are necessary to achieve this rate (unless very good starting values are used).

To overcome these difficulties, a few steps of IVIT is used first to obtain a better approximate pair (λ, x), then a procedure called SUBDIM is used to determine if λ is an isolated eigenvalue. When λ is isolated, we switch to RQI, and (λ, x) provided by IVIT should be closer to the cubic converging range of RQI. Otherwise, from SUBDIM, the dimension of an invariant subspace corresponding to eigenvalues close to λ and an approximate basis for the subspace are available. Then further improvement of the subspace to the required accuracy can be achieved by using SIRR.

This is described in detail in the following paragraphs.

INVIT: The inverse iteration we use is a slight modification of the ordinary one:

(0) Set $x^{(0)} = x_i(t_j)$, $\mu = \tilde{\lambda}_i(t_{j+1})$, $k = 0$.
(1) Solve

$$(A(t_{j+1}) - \mu I)\, y^{(k)} = x^{(k)} \tag{20}$$

for $y^{(k)}$.

(2) Compute $\gamma_k = \| y^{(k)} \|$ and let

$$x^{(k+1)} = \frac{y^{(k)}}{\gamma_k}.$$

(3) If $\gamma_k \leqslant M$ and $\gamma_k/\gamma_{k-1} \geqslant \alpha$ then $k = k + 1$, go to (1)
else let $\lambda = \rho(x^{(k+1)})$, $x = x^{(k+1)}$, quit,
where

$$\rho(x^{(k+1)}) = (x^{(k+1)})^{\mathrm{T}} A(t_{j+1}) x^{(k+1)}$$

is the Rayleigh quotient.

There are two control parameters in this procedure, namely M and α. In our algorithm, we set $M = 10^4$ and $\alpha = 1.2$. It is well known that $x^{(k)}$ will converge to an invariant subspace corresponding to the eigenvalues close to μ and

$$\gamma_k \to \frac{1}{\min|\mu - \lambda_i|}.$$

The convergence comes in two ways: if the prediction μ is very close to some true eigenvalues ($\min|\mu - \lambda_i| < 1/M$), $\gamma_k > M$ is quickly satisfied; on the other hand, if μ is not close to any particular eigenvalue ($\min|\mu - \lambda_i| > 1/M$), then we must wait for the second condition $\gamma_k/\gamma_{k-1} < \alpha$ to be satisfied. Inequality $\gamma_k/\gamma_{k-1} < \alpha$ indicates that γ_k will not grow significantly in the following iterations. In this case, the inverse iteration procedure has been stabilized or converged. For our purpose, this procedure is not affected by close eigenvalues, since one of the two conditions must be satisfied at the end. There are other choices for M and α. Since we will switch to a faster method eventually, it is appropriate to impose a loose convergence condition.

Before the decision with regard to which method to use next (RQI or SIRR), we must find out the dimension of the invariant subspace corresponding to the eigenvalues close to λ. To this end, we introduce SUBDIM.

Let $\{y_1, y_2, \ldots, y_n\}$ be a set of linearly independent vectors chosen beforehand. Take y_1, orthogonalize it against x and label it y_1 again. Then

(0) let $k = 1$, $z_1 = x$
(i) solve

$$(A(t_{j+1}) - \lambda I)z = y_k \tag{21}$$

for z
(ii) compute $\| z \|$.
if $\| z \|$ is large, then
let $z_{k+1} = z/\| z \|$, take y_{k+1}, use MGS to orthogonalize it against $\{z_1, z_2, \ldots, z_{k+1}\}$,
let iterat = 1, $k = k + 1$, goto (i).
else
if iterat = 1, orthogonalize z against $\{z_1, z_2, \ldots, z_k\}$, let $y_k = z/\| z \|$,
iterat = iterat + 1, goto (i).
end if.

Here MGS stands for modified Gram–Schmidt process [11].

Out of this procedure, k is the dimension and $\{z_1, z_2, \dots, z_k\}$ is an approximate basis for the invariant subspace corresponding to the eigenvalues close to λ.

The theoretical background for this procedure is the following.

Suppose $\lambda_1, \lambda_2, \dots, \lambda_m$ are in a cluster for some $1 \leqslant m < n$ such that

$$\max_{1 \leqslant l \leqslant m} |\lambda_l - \lambda| \leqslant \varepsilon, \qquad \min_{m+1 \leqslant l \leqslant n} |\lambda_l - \lambda| \geqslant \sigma > \varepsilon. \tag{22}$$

Expand the vector y_k in (21) in terms of $x_1, x_2, \dots, x_n$:

$$y_k = \sum_{l=1}^{n} a_l^{(k)} x_l = \sum_{l=1}^{m} a_l^{(k)} x_l + \sum_{l=m+1}^{n} a_l^{(k)} x_l \overset{\text{def}}{=} y_k^{(0)} + y_k^{(1)},$$

then the solution of (21) is

$$\begin{aligned} z &= (A(t_{j+1}) - \lambda I)^{-1} y_k \\ &= (A(t_{j+1}) - \lambda I)^{-1} \sum_{l=1}^{n} a_l^{(k)} x_l \\ &= \sum_{l=1}^{n} \frac{a_l^{(k)}}{\lambda_l - \lambda} x_l \\ &= \sum_{l=1}^{m} \frac{a_l^{(k)}}{\lambda_l - \lambda} x_l + \sum_{l=m+1}^{n} \frac{a_l^{(k)}}{\lambda_l - \lambda} x_l \\ &\overset{\text{def}}{=} z^{(0)} + z^{(1)}. \end{aligned}$$

Now,

$$\|z^{(1)}\| = \sqrt{\sum_{l=m+1}^{n} \frac{(a_l^{(k)})^2}{|\lambda_l - \lambda|}} \leqslant \frac{\sqrt{\sum_{l=1}^{n} (a_l^{(k)})^2}}{\min_{m+1 \leqslant l \leqslant n} |\lambda_l - \lambda|} \leqslant \frac{1}{\sigma}.$$

If $k < m$, y_k is orthogonal to $\tilde{V}_k = \operatorname{span}\{z_1, z_2, \dots, z_k\}$, hence y_k has a nontrivial component $y_k^{(0)}$ in $V_m = \operatorname{span}\{x_1, x_2, \dots, x_m\}$, and the norm of z grows rapidly during the iteration. In fact,

$$\|z\| \geqslant \|z^{(0)}\| = \sqrt{\sum_{l=1}^{m} \frac{(a_l^{(k)})^2}{|\lambda_l - \lambda|}} \geqslant \frac{1}{\varepsilon} \|y_k^{(0)}\|.$$

Hence $\|z\|$ becomes large in one or two iterations, and as the procedure continues, the subspace dimension continues to grow. When $k = m$, however, y_k is orthogonal to the whole subspace $\tilde{V}_m$ which is a good approximation to V_m, hence $y_k^{(0)} \approx 0$, and $\|z\|$ becomes small, we then reach the end of the procedure. In such a way, the dimension of the cluster can be determined and a good approximate basis for the corresponding invariant subspace becomes available.

It is necessary to be specific about "large" and "small" when this algorithm is used. Notice that iterating (21) continuously yields

$$\|y_k^{(0)}\| \to 1, \qquad \|y_k^{(1)}\| \to 0,$$

hence, at the end,

$$\|z\| \geqslant \frac{1}{\varepsilon}.$$

Therefore, $\|z\|$ may be categorized as large if it is greater than $1/2\varepsilon$, it is not large otherwise, where ε is as in (22). Our experience is that, when the dimension of a cluster is slightly overestimated, it causes minimum effect, but when it is underestimated, the subspace iteration may be slowed down considerably. One remedy for the possible incorrect dimension count is that the "dynamic" scheduling (described later) may be employed during the progress of the subspace iteration.

Now we are able to decide whether to use RQI or SIRR for our final iterations. Namely, if $k = 1$ from SUBDIM, QRI is used, otherwise SIRR is called.

4.5. Dynamic subspace iteration

When the convergence of the iterations is too slow (more than three iterations, for instance), it usually indicates that the estimate of the subspace dimension is inappropriate, either too big or too small. If the iteration process is monitored carefully, the subspace dimension can be adjusted during the progress when it becomes necessary: after a few steps of Rayleigh–Ritz procedure [24], only those Ritz vectors with their Ritz values close together are kept for further iterations. The subspace dimension is then reduced and the condition of the subspace is improved. This in turn will reduce the computation and speed up the convergence. If, on the other hand, all the Ritz values are close and convergence is still slow, then it becomes necessary to enlarge the subspace. This can be done by calling procedure SUBDIM to determine a more accurate dimension, and the subspace iteration resumes after this. Such a process often accelerates the convergence considerably.

Remark 4.1. For a large sparse matrix, factorization is feasible, but it is still one of the major parts of the computation when solving an equation. Therefore we keep the number of factorizations to a minimum in our algorithm by including procedures such as INVIT, SUBDIM, SIRR, and RQI. INVIT, SUBDIM, and SIRR are all iterative methods. They only require solving the systems with different right-hand sides, that is, new factorization is not needed for every iteration step. These methods are not the fastest, but by combining them with faster methods such as RQI, we can still achieve a high convergence rate while reducing the computation time.

4.6. Checking

When RQI or SIRR is performed, equations such as (20) must be solved. By taking advantage of the symmetry, the so-called symmetric Gauss elimination method [2] can be applied and the inertia of $A - \mu I$ is readily available from the $L^{\mathrm{T}}DL$ decomposition [2, 7]. With this information, we can identify to which eigenvalue the iteration converges. Usually, jumping to an unwanted curve does not occur when eigenvalue separations are good. When the separations are poor, our SIRR will provide a cluster of eigenvalues which will usually include the one we seek. With all the efforts in the correction step, our iteration is expected to be successful. There are occasions, however, when a wrong one is obtained, but still this may not be a total waste since it may very well

be the case that this eigenvalue has not yet been found by other processors. Usually, the number of eigenvalues desired is larger than the number of processors available, so the chance of a waste is small.

4.7. *Clearing up*

During the tracing of eigencurves, a significant amount of time is spent on keeping the process on the right curve. There are several existing techniques for this purpose [17, 18]. However, there are cases where these techniques are too costly. Several new approaches have been introduced in our algorithm. They are designed especially for large matrices. Instead of imposing very strong restrictions on those control parameters, our strategy is to abandon the process if convergence to a desired value is not observed after reasonable efforts have been made. In the end, most of the needed eigenvalues of A are computed. A few of those missed are scattered across the spectrum of A and isolated by those found, that is, there are several small intervals containing those missed values. By counting the inertia corresponding to the end points of each interval, the number of eigenvalues inside such an interval is known. These values are well separated from the neighboring ones since clusters have been found by SUBDIM in the algorithm. Under these favorable conditions, the subspace iteration method is applied in each interval efficiently to find the eigenvalues inside it, and this can be done in parallel.

4.8. *Step size control*

Our extensive numerical experience shows that the step size should be chosen as large as possible. Allowing small step size can lead to inefficiency. In our algorithm, the first attempt of the step size h is chosen to be $\max\{0.25, \frac{1}{2}(1-t)\}$. If convergence is not achieved at some step and $h > 0.25$, the step size is cut in half and the process is repeated. If the failing occurs when $h \leqslant 0.25$, we simply abandon the process of following this specific curve. Then the eigenvalue may be computed by some other process (such as SUBDIM if it is among a cluster) or may be considered missed. Because we have a back up procedure for those eigenvalues missed, this strategy would cause no problem, instead, it is more efficient to have a lower limit imposed on h. There are cases where under no control limit, h becomes very small, hence causing the long execution time simply because there is a single "bad point". In our many examples, the largest number of missed eigenvalues is no more than 10% of the total number of eigenvalues sought.

5. Numerical experiments

Our algorithm has been implemented on several machines using FORTRAN. The sparse matrix A is stored using the *coordinate scheme* [7, pp. 23–24], that is, the matrix is specified as a set of triples (a_{ij}, i, j), they are stored in one real array and two integer arrays. Because A is symmetric, only the upper triangular part is stored.

For solving the sparse linear system

$$(A(t) - \lambda I)y = x, \tag{23}$$

the MA27 subroutines from the *Harwell Subroutine Library* are used. This code can solve indefinite symmetric systems, such as (23), stably and with minimum overhead above the code for positive definite systems. It can also provide the inertia of $A(t) - \lambda I$ as a by-product. Both are vitally important to our algorithm.

MA27 has three separate steps: *Symbolic Analyse*, *Factorize*, and *Solve*. *Symbolic Analyse* exploits the sparse structure of the matrix and estimates the working space needed for later steps. *Factorize* implements a version of Gauss Elimination to compute the LDL^T decomposition of the matrix. The last step, *Solve*, uses this decomposition to actually solve the matrix system. Since the underlying matrices $A(t)$ have the same sparse structure for all $t \in [0,1]$, *Symbolic Analyse* is required only once in the whole algorithm. In addition, for inverse iterations, one needs to solve systems with different right-hand sides only, new LDT^T decompositions are not necessary, i.e., only the last step, *Solve*, is called for each iteration. This leads to significant savings in time because *Factorize* is the most expensive step of the three.

5.1. Test matrices

Three types of matrices are used to test our algorithm. The first two can be found in [4].

1. *Diagonally-disordered matrices.* These matrices arise in the study of two-dimensional $NX \times NY$ arrays of atoms in disordered systems [14]. The associated matrix A has order $N = NX \times NY$,

$$A = \begin{bmatrix} B & I & & I \\ I & B & \ddots & \\ & \ddots & \ddots & I \\ I & & I & B \end{bmatrix}, \qquad B = \begin{bmatrix} x & 1 & & 1 \\ 1 & x & \ddots & \\ & \ddots & \ddots & 1 \\ 1 & & 1 & x \end{bmatrix}.$$

The matrix A is almost block tridiagonal. I is the identity matrix of order NX. Each B block is $NX \times NX$ and there are NY blocks down the diagonal of A. The diagonal elements of B are randomly generated numbers. A scaling parameter bounds the magnitudes of these disordered terms. In the simplest case, these entries are chosen from a uniform random distribution over an interval [− SCALE, SCALE] determined by the user-specified scaling parameter SCALE. A second class of diagonally-disordered matrices is obtained by choosing the diagonal elements randomly as either 0 or SCALE. If NX and NY are relatively prime, then all of the eigenvalues of A are distinct.

2. *Poisson matrices.* When the Laplace equation

$$u_{xx} + u_{yy} = \lambda u, \quad R = \{(x,y) \mid 0 \leqslant x \leqslant X,\ 0 \leqslant y \leqslant Y\}$$

is solved numerically, the differential equation is replaced by an algebraic linear system, $Ax = \lambda x$, obtained from discretizing the Laplace operator on the rectangle. The order of the resulting matrix is $N = NX \times NY$ where NX is the number of subdivisions in x-direction and NY is the number of

subdivisions in y-direction. The matrix A is block tridiagonal,

$$A = \begin{bmatrix} B & C & & & \\ C & B & \ddots & & \\ & \ddots & \ddots & C & \\ & & C & B & sC \\ & & & sC & B \end{bmatrix}, \quad B = \begin{bmatrix} 1 & -sc & & & & \\ -sc & 1 & -c & & & \\ & -c & \ddots & \ddots & & \\ & & \ddots & \ddots & -c & \\ & & & -c & 1 & -sc \\ & & & & -sc & 1 \end{bmatrix}.$$

The parameter c is user-specified and must be chosen such that $0 < c \leqslant 0.5$. Here, $C = -(0.5 - c)I$. For Dirichlet boundary conditions $s = 1$, and for Neumann conditions $s = \sqrt{2}$. Under Dirichlet conditions, the eigenvalues (and eigenvectors) of A are known,

$$\lambda_{ij} = [1 - 2c\cos(\pi i/(NX + 1)) - 2(0.5 - c)\cos(\pi j/(NY + 1))], \quad 1 \leqslant i \leqslant NX, \ 1 \leqslant j \leqslant NY.$$

Under Neumann conditions, however, the eigenvalues and eigenvectors are not known a priori. By varying the value of c, many different eigenvalue distributions can be obtained.

3. *Random matrices.* The sparse matrices in this group are generated in three steps: Let N be the order of the matrix. We use *rand* to represent random number generator that produces uniform distribution on interval (0, 1). Then

(0) Clear the matrix:
  for $I = 1, \ldots, N$; $J = I, \ldots, N$
  $A(I,J) = 0.0$.

(1) Generate N entries randomly in the upper triangular part of A:
  for $i = 1, \ldots, N$
  $I = \text{Int}(N * \text{rand}(i) + 1)$, $J = \text{Int}(N * \text{rand}(I) + 1)$
  if $I < J$ *then* $A(I,J) = \text{rand}(i) + A(I,J)$ *else* $A(J,I) = \text{rand}(i) + A(J,I)$.

(2) Generate N entries randomly along upper 5 diagonals:
  for $i = 1, \ldots, N$
  $I = \text{Int}(N * \text{rand}(i) + 1)$, $J = \text{Min}\{N, I + \text{Int}(5 * \text{rand}(I))\}$
  $A(I,J) = \text{rand}(i) + A(I,J)$.

(3) Generate N entries randomly along the main and upper diagonal:
  for $i = 1, \ldots, N$
  $I = \text{Int}(N * \text{rand}(i) + 1)$, $J = \text{Min}\{N, I + \text{Int}(2 * \text{rand}(I))\}$
  $A(I,J) = \text{rand}(i) + A(I,J)$.

A matrix generated this way has nonzero entries concentrated near the main diagonal, a pattern shared by many sparse matrices in applications.

5.2. *Results on IBM 3090 vector machine*

IBM 3090 VF system is a machine with vector extensions to the IBM System/370 XA and System/370 architectures. It has one vector Facility (vector length 128), 65 megabytes central memory, 16 channels and VM/SP operating system.

Table 1
Test data from IBM

Matrix	Order	Homotopy	Lanczos	Ratio
	500	85.34	43.91	0.515
Disordered	1000	181.01	94.49	0.522
	5000	1198.72	631.73	0.527
	500	65.29	37.44	0.573
Poisson	1000	133.54	75.61	0.579
	5000	676.41	394.35	0.583
	500	46.83	23.65	0.505
Random	1000	94.66	49.04	0.518
	5000	481.37	253.68	0.527

We compare our algorithm with EA15 — an algorithm for eigenproblems using the Lanczos method. This algorithm is made up of several subroutines in *Harwell Subroutine Library*. The user supplies an interval which contains all the eigenvalues of interest and the algorithm finds all eigenvalues inside the interval and the corresponding eigenvectors.

Table 1 contains a list of execution times for computing the first 50 eigenpairs for different matrices. It is not a total surprise that our algorithm runs behind the Lanczos algorithm in sequential machine IBM 3090, since we are aiming for an efficient parallel algorithm.

Nevertheless, even on such a sequential machine, our algorithm can outperform Lanczos' in several important situations.

(1) *Interior eigenvalues*: The above comparison with EA15 are based on the computation of some external eigenvalues. It is well known that Lanczos algorithm is most efficient for such problems. However, in many applications, interior eigenvalues are desired. In such circumstances, our homotopy method is much faster than the Lanczos algorithm. The reason is that, during Lanczos iterations, extremal eigenvalues usually emerge (no matter they are needed or not) before the interior ones.

Table 2 shows the execution times for the computation of 20 middle eigenpairs of the matrices listed.

(2) *Better initial matrix*: As mentioned in Remark 2.7, for a typical application problem, one needs to solve a series of matrix eigenvalue problems. When this situation occurs, using a matrix in the sequence as the initial matrix, we expect to have a much better result than using the block diagonal initial matrix. An experiment is constructed to illustrate this: randomly choose $\frac{1}{2}n$ nonzero entries in A, perturb them by small random numbers in $(-0.05, 0.05)$. Then use the perturbed matrix as the initial matrix D. Table 3 shows the execution times for both algorithms — homotopy and EA15. The time needed for the computation of initial eigensystems is not included in the homotopy algorithm since it is assumed that eigensystems for that matrix are obtained in the previous step of the sequence.

Remark. The Lanczos algorithm can also take some advantage of such a situation by using one of the computed eigenvectors of our initial matrix as the starting vector to generate the Krylov subspaces (or use several to start a block Lanczos algorithm).

Table 2
Interior 20 eigenpairs on IBM 3090

Matrix	Order	Homotopy	EA15	Ratio
	500	42.68	86.79	2.03
Disordered	1000	82.38	332.07	4.03
	5000	354.10	2487.48[a]	7.02
	500	33.52	70.73	2.11
Poisson	1000	63.91	270.34	4.23
	5000	347.64	2471.72[a]	7.11
	500	24.78	55.75	2.25
Random	1000	49.46	214.16	4.33
	5000	251.47	1750.23[a]	6.95

[a] Eigenvalues only.

Table 3
First 50 eigenpairs on IBM 3090 with perturbed starting matrix

Matrix	Order	Homotopy	EA15	Ratio
	500	11.96	43.91	3.67
Disordered	1000	33.39	94.49	2.83
	5000	199.28	631.73	3.17
	500	11.52	37.44	3.25
Poisson	1000	18.72	75.61	4.04
	5000	119.14	394.35	3.31
	500	9.98	23.65	2.37
Random	1000	23.69	49.04	2.07
	5000	85.70	253.68	2.96

5.3. *Results on BBN butterfly parallel machine*

BBN butterfly machine is a share-distributed memory architecture with 96 MC68020/MC68882 BBN GP1000 processors, the maximal number of processors available to users is 90. The results in Tables 4 and 5 are obtained from averaging the times used in computing the first 100 eigenpairs of 1000 by 1000 diagonally-disordered matrix, Poisson matrix, and random matrix.

Speed-up in Table 4 measures the parallelization level of the algorithm itself. We achieve close to perfect speed-up when the number of processors is less than or equal to 16 (15 times faster using 16 processors). Moreover, the efficiency (speed-up/number of processors) can be further improved if the problem is of sequential type (Remark 2.7), the "bottle neck" of computing the initial eigensystems will disappear. Another possible way of improving our algorithm is to apply the

Table 4
Speed-up for homotopy algorithm

Number of processors	1	2	4	8	16	32	64
Time for homotopy	2186.85	1095.73	565.08	282.90	145.61	88.14	74.08
Speed-up	1	1.99	3.87	7.73	15.02	24.81	29.52

Table 5
Speed-up over EA15

Number of processors	1	2	4	8	16	32	64
Time for homotopy	2186.85	1095.73	565.08	282.90	145.61	88.14	74.08
Time for EA15	998.17	998.17	998.17	998.17	998.17	998.17	998.17
Speed-up	0.46	0.91	1.77	3.41	6.86	11.33	13.47

Table 6
The residuals of eigenpairs

Matrix	Order	Homotopy	Lanczos
Disordered	500	$0.99 \cdot 10^{-15}$	$0.81 \cdot 10^{-15}$
	1000	$0.13 \cdot 10^{-14}$	$0.94 \cdot 10^{-15}$
	5000	$0.27 \cdot 10^{-14}$	$0.13 \cdot 10^{-14}$
Poisson	500	$0.32 \cdot 10^{-14}$	$0.74 \cdot 10^{-15}$
	1000	$0.51 \cdot 10^{-14}$	$0.79 \cdot 10^{-15}$
	5000	$0.76 \cdot 10^{-14}$	$0.11 \cdot 10^{-14}$
Random	500	$0.27 \cdot 10^{-15}$	$0.18 \cdot 10^{-15}$
	1000	$0.22 \cdot 10^{-15}$	$0.20 \cdot 10^{-15}$
	5000	$0.23 \cdot 10^{-15}$	$0.39 \cdot 10^{-15}$

homotopy method recursively — using the idea of "Divide and Conquer". Nevertheless, our present algorithm can achieve high speed-up close to 30.

Table 5 lists the speed-up of our algorithm over EA15. Notice that our algorithm can be more than 13 times faster than Lanczos' — one of the best sequential algorithms.

5.4. *Accuracy and orthogonality*

The accuracy of our algorithm is also very satisfactory compared to EA15. Table 6 lists the maximal residuals

$$\max_{1 \leqslant i \leqslant 50} \| Ax_i - \lambda x_i \|$$

for the computed eigenpairs from these two algorithms. The data used here are collected from IBM 3090. The residual measures the accuracy for both eigenvalue and eigenvector.

Table 7
The orthogonality of eigenvectors

Matrix	Order	Homotopy	Lanczos
Disordered	500	$0.36 \cdot 10^{-15}$	$0.52 \cdot 10^{-15}$
	1000	$0.66 \cdot 10^{-15}$	$0.71 \cdot 10^{-15}$
	5000	$0.67 \cdot 10^{-15}$	$0.82 \cdot 10^{-15}$
Poisson	500	$0.27 \cdot 10^{-14}$	$0.75 \cdot 10^{-15}$
	1000	$0.23 \cdot 10^{-14}$	$0.84 \cdot 10^{-15}$
	5000	$0.59 \cdot 10^{-14}$	$0.20 \cdot 10^{-14}$
Random	500	$0.87 \cdot 10^{-16}$	$0.13 \cdot 10^{-15}$
	1000	$0.19 \cdot 10^{-15}$	$0.36 \cdot 10^{-15}$
	5000	$0.25 \cdot 10^{-15}$	$0.47 \cdot 10^{-15}$

Table 7 compares the orthogonality among the computed eigenvectors. Because A is symmetric, these eigenvectors are orthogonal in theory, i.e., $\| X^{\mathrm{T}}X - I \| = 0$. Listed in the last two columns are the maximal entries of the computed matrices $X^{\mathrm{T}}X - I$ from these two algorithms.

References

[1] P. Arbenz and G.H. Golub, On the spectral decomposition of hermitian matrices modified by low rank perturbations with applications, *SIAM J. Matrix Anal. Appl.* **9**(1) (1988) 40–58.

[2] J.R. Bunch and B.N. Parlett, Direct methods for solving symmetric indefinite systems of linear equations, *SIAM J. Numer. Anal.* **8**(4) (1971) 639–655.

[3] E. Chu, A. George, J. Liu and E. Ng, Sparspack: Waterloo sparse matrix package user's guide for sparspack-a, Report CS-84-36, Department of Computer Science, Univ. of Waterloo, Ontario, Canada, 1984.

[4] J.K. Cullum and K.A. Wiloughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. I: Theory* (Birkhäuser, Boston, 1985).

[5] J.J.M. Cuppen, A divide and conquer method for the symmetric eigenproblem, *Numer. Math.* **36** (1981) 197–195.

[6] J.J. Dongarra and D.C. Sorensen, A fully parallel algorithm for the symmetric eigenvalue problem, *SIAM J. Sci. Statist. Comput.* **8**(2) (1987) 139–154.

[7] I.S. Duff, A.M. Erisman and J.K. Reid, *Direct Methods for Sparse Matrices* (Clarendon Press, Oxford, 1986).

[8] S.C. Eisenstat, M.C. Gursky, M.H. Schultz and A.H. Sherman, Yale sparse matrix package, 1: The symmetric codes, *Internat. J. Numer. Methods Engrg.* **18** (1982) 1145–1151.

[9] K. Fan, Maximum properties and inequalities for eigenvalues of completely continuous operators, *Proc. Nat. Acad. Sci. USA* **75** (1951) 760–766.

[10] A. George and J.W.H. Liu, *Computer Solution of Large Sparse Positive-Definite Systems* (Prentice-Hall, New York, 1981).

[11] G.H. Golub and C.F. Van Loan, *Matrix Computations* (Johns Hopkins University Press, Baltimore, MD, 1983).

[12] P.S. Jenson, The solution of large symmetric eigenproblems by sectioning, *SIAM J. Numer. Anal.* **9** (1972) 534–545.

[13] T. Kato, *Perturbation Theory for Linear Operators* (Springer, New York, 1966).

[14] S. Kirkpatrick and T.P. Eggarter, Localized states of a binary alloy, *Phys. Rev. B* **6** (1972) 3589–3600.

[15] K. Li and T.Y. Li, An algorithm for symmetric tridiagonal eigenproblems – divide and conquer with homotopy continuation, *SIAM J. Sci. Comput.* **14**(3) (1993) 735–751.

[16] T.Y. Li, Z. Zeng and L. Cong, Solving eigenvalue problems of real nonsymmetric matrices with real homotopies, *SIAM J. Numer. Anal.* **29**(1) (1992) 229–248.

[17] T.Y. Li and N.H. Rhee, Homotopy algorithm for symmetric eigenvalue problems, *Numer. Math.* **55**(3) (1989) 265–280.
[18] T.Y. Li and Z. Zeng, Homotopy-determinant algorithm for solving nonsymmetric eigenvalue problems, *Math. Comput.* **59**(200) (1992) 483–502.
[19] T.Y. Li, H.Z. Zhang and X.H. Sun, Parallel homotopy algorithm for symmetric tridiagonal eigenvalue problem, *SIAM J. Sci. Statist. Comput.* **12**(3) (1988) 155–165.
[20] S.S. Lo, B. Philippe and A. Sameh, A multiprocessor algorithm for the symmetric eigenvalue problem, *SIAM J. Sci. Statist. Comput.* **8**(2) (1987) 155–165.
[21] S.C. Ma, M.L. Patrick and D.B. Szyld, A parallel, hybrid algorithm for the generalized eigenproblem, preprint, 1988.
[22] Murata and Horikoshi, A new method for the tridiagonalization of the symmetric band matrix, *Inform. Process. in Japan* **15** (1975) 108–112.
[23] J. Ortega, *Numerical Analysis; A Second Course* (Academic Press, New York, 1972).
[24] B.N. Parlett, *The Symmetric Eigenvalue Problem* (Prentice-Hall, Englewood Cliffs, NJ, 1980).
[25] B.T. Smith, J.M. Boyle, J.J. Dongarra, B.S. Garbow, Y. Ikebe, V.C. Klema and C.B. Moler, *Matrix Eigensystem Routines – EISPACK Guide*, Lecture Notes in Computer Science **6** (Springer, Berlin, 2nd ed., 1976).
[26] D. Szyld, Criteria for combining inverse and rayleigh quotient iterations, *SIAM J. Numer. Anal.* **25**(6) (1988) 1369–1375.
[27] J.H. Wilkinson, *The Algebraic Eigenvalue Problem* (Oxford University Press, Oxford 1965).